CONNEXIONS™

The Interoperability Report

May 1989 Volume 3, No. 5

ConneXions—
The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, CA 94040, USA. Phone: 415-941-3399.

ISSN 0894-5926

From the Editor

This month we continue our series *Components of OSI* with an overview of X.400, the ISO/CCITT standard for message handling systems. The article is by Julian Onions of the University of Nottingham, England. Electronic mail is by far the most used application in today's internets, and will undoubtedly continue to be so in the networks based on OSI as they emerge, particularly since X.400 offers a much richer set of document types which can be exchanged in the message. Julian describes the X.400 model, and compares the features to our existing mail protocols.

As OSI becomes more prevalent, there will be a need for mail gateways to bridge the gap between the millions of users of existing systems (many of them RFC 822 based) and future X.400 systems. This issue is addressed in RFC 987 ("Mapping between X.400 and RFC 822"), and there are several projects underway to implement such gateways. *ConneXions* will be reporting on these projects in future issues.

Interoperability means more than having a common set of network protocols and base applications. There has been much interest in Sun Microsystems' *Network File System* (NFS) as well as the MIT *X-Windows* application. Vendors gathered together in Santa Clara, California in mid-February of this year to take part in a week long interoperability testing session called *Connectathon '89*. We bring you a brief report on page 9.

Interoperability could also be defined as the transparent access to software and hardware resources across networks. Such access raises questions about copyright and licensing of network-based utilities. Jim Geismann and Tom Wood from Marketshare, Inc. discuss these matters in an article about *Network License Servers* starting on page 10.

Finally we list some upcoming events which should be of interest to our readers.

A reminder that all back issues of *ConneXions* are available for purchase. Write *ConneXions*, 480 San Antonio Road, Suite 100, Mountain View, CA 94040 for free index pages covering Volume 1 (1987) and Volume 2 (1988).

Components of OSI: The X.400 Message Handling System

by Julian Onions, University of Nottingham

What is X.400?

X.400 is the international standard for message handling systems, defined jointly by ISO [1] and CCITT [2]. It is sometimes the case that an international standard is defined by taking an existing ad-hoc standard and cleaning it up slightly. However, with X.400 the standards bodies have come up with something outwardly very different from current mail formats.

Early input to X.400 was provided groups like IFIP WG-6.5 whose members were experienced in electronic mail. This work was then expanded and refined to produce a set of recommendations and an ISO multi-part standard. There are two versions of the CCITT recommendations—the 1984 version and the 1988 version. The CCITT work in 4 year periods, and the 1988 version is a substantially revised version of the 1984 version. The CCITT recommendations consists of:

X.400	System and Service Overview
X.401	Basic Service Elements and Optional User facilities
X.402	Overall Architecture
X.407	Abstract Service Definition Conventions
X.408	Encoded Information type Conversion Rules
X.411	Abstract Service Definition and Procedures
X.413	Message Store Abstract Service
X.419	Protocol Specifications
X.420	Interpersonal Messaging System
X.430	Access Protocols for Teletex Terminals

The X.400 Model

X.400 defines a model for electronic mail which has essentially three layers, as shown in Figure 1. The top most layer is the user who performs actions such as composing and filing mail. At the next level is the *User Agent* (UA). This is a piece of software or hardware that aids the user in composing and receiving mail. Its task is to provide the user with access to the message handling system. All interaction with the system goes through this interface and all messages are received via this interface (at least in the 1984 model). At the bottom is the Message Transfer Layer (MTL). This is an interconnection of *Message Transfer Agents* (MTA) which route messages to their destination.

The model has some analogies with the postal system. Users exchange *messages*, consisting of several possible *body parts* of different formats. These body parts are the message data which the user wishes to post. The message contains headers specifying information about its content. This includes the message recipients, the subject and other information. Then, when posted, the message is encapsulated in an *envelope*. The envelope contains all the information relevant for the delivery of the message. This includes the recipient, the originator and sufficient information about the message being carried to allow delivery.

It is an important distinction that the envelope and the contents are entirely separate. This allows MTAs to transfer the message as an opaque object. Indeed, the content need not be a valid message to be transported by the MTAs.

The message headers and body types are defined by the P2 protocol. The envelope is defined by the P1 protocol. The P2 protocol provides a protocol between the two (or more) communicating user agents. The P1 protocol provides the inter-MTA level protocol and contains all the information pertinent to delivery of the message. There are two other protocols worth noting in passing. The P3 protocol allows the UA to be remote from the MTA and to submit and receive messages in batch mode. This might be used by a UA connected to an MTA by a one way link, to pick up and post messages. The P7 protocol is similar but connects to a message store where messages can be browsed, forwarded, deleted, listed etc.

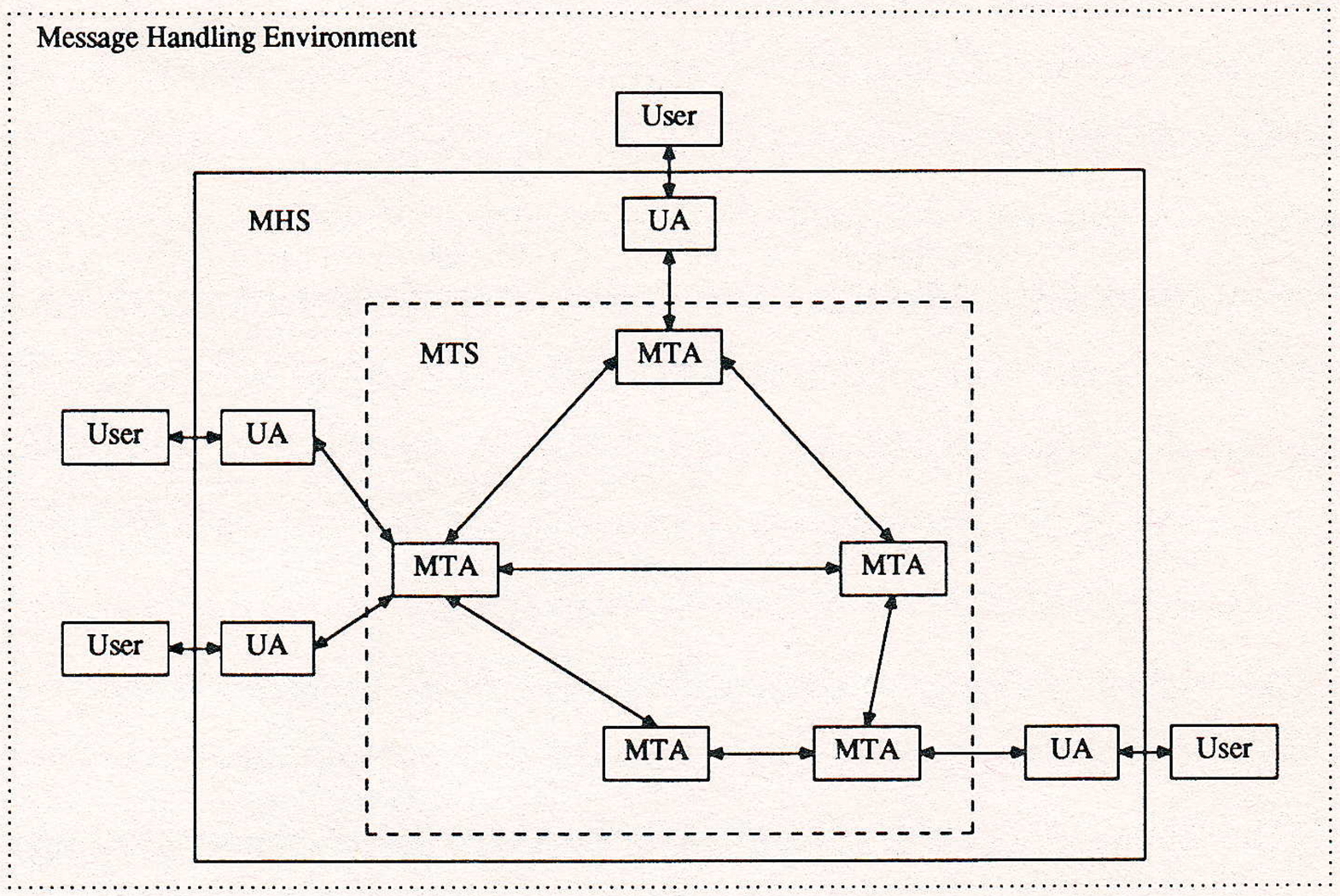


Figure 1: The MHS Model

X.400 Adressing

X.400 has its own type of address, designated the *O/R-address* (Originator/Recipient address). An O/R-address is a binary encoding of attribute value pairs, and several forms are possible. An example X.400 O/R address might look like:

Country	GB
Administrative Management Domain	Gold 400
Private management Domain	UK.AC
Organisation	Nottingham University
Organisation Unit	Computer Science
Personal Name/Given	Julian
Personal Name/Surname	Onions

This format contains several items of information which are concerned with routing rather than naming. The management domains for instance are present mainly for allowing connections between separately administered domains to be resolved.

In the 1988 specification, use may be made of the Directory service (described in a companion article to this one) for the naming of users. The Directory will take a much looser description and resolve it to an individual (or give a list of matching people).

continued on next page

The X.400 Message Handling System *(continued)*

In this case the sort of name you would enter would be more like:

Country	GB
Organisation	Nottingham University
Common Name	Julian P. Onions

The Directory would then resolve this to a Directory entry (or perhaps ask for more information if the name was ambiguous) and retrieve the users address X.400 address. In 1988 X.400 an address may be submitted as either the full O/R-address, as the Directory name, or both. If the O/R-address is missing when the message is submitted to an MTA, the MTA will map the Directory name to the O/R-address using the Directory. However, this task may equally well be performed in the UA allowing interactive action such as Directory searching to find the correct recipient.

The user-level protocol—P2

The P2 protocol determines the nature of end-to-end communication between users. This includes defining message header fields such as the "To" field, the "Subject" and so on. The P2 message header has a binary encoding defined in terms of ASN.1 [3] which is a powerful abstract data description language used throughout OSI. The elements that are standard in the P2 protocol are:

A message identifier
An originator
A list of authorising users
A list of primary recipients
A list of copy recipients
A list of Blind copy recipients
A reference to a message to which this is a reply
A list of messages that this message obsoletes
A list of cross referenced messages
A subject
An expiry date
A time by which a reply should be sent by
A list of users to reply to
An importance field
A sensitivity field
An indication if this message has been automatically forwarded

The body of the P2 message is also structured and is a sequence of body parts. This allows mixing of numerous formats in the same message. Some of the body types defined within X.400 are IA5 text (ASCII to you and me), Telex, Fax and Voice. A particularly useful sort of body type is the "forwarded message." This is a recursive type allowing a complete message to be part of another message. This is the way in which X.400 supports the forwarding of messages.

P22

The 1988 version of P2 (called P22) is a superset of the 1984 P2 and allows for extensions of the protocol in a flexible way. There is an "extension body" type, which allows any format required and contains, as part of its structure, an identifier to say what type it is. However, this is a double edged sword. Its strength is that anyone can define a new body type (e.g., PostScript or TeX) and so can exchange any format of message. Its weakness is that the receiver may not be able to interpret these new types. In addition to the body extensions, there is a new P2 header which specifies the language in use in the message.

There is also a general extension type which allows the defining of any new header type in a similar way to new body types.

The transport protocol—P1

The P1 protocol contains all the information necessary for MTAs to relay the message from its source to its destination. This includes:

A message identifier
The Originator
The content type
The types of information carried
The priority
Some flags
A delivery time
A list of recipients
Some trace information

The *message identifier* is used to identify this message throughout its lifetime. It does not identify the message content as such, just this attempt to send the message. The P2 level message identifier identifies the actual message content.

The *Originator* is the O/R address of the user who submitted the message. This may be different to the P2 O/R address if, for instance, the message is being submitted on behalf of someone else. Error reports will be directed to this address, but P2 replies may not.

The *types of information carried* is used by the MTA for determining UA capabilities. This is important when many formats are carried. For example, it makes no sense to deliver a voice message to a telex terminal without some conversion being done!

The *priority* is an indication of the priority of a message. It is presumed that high priority messages may be processed faster but cost more. The flags mark whether the message may be converted for delivery, if the contents should be returned and so on.

The *delivery time* is used for delayed delivery. It is possible to send a message and specify that it should be delivered at some time in the future (possibly useful for birthday messages!).

The *recipients* is a list of O/R addresses coupled with some flags. The flags allow the requesting of facilities like confirmed delivery, and indicate whether explicit conversion of contents should be attempted, on a per recipient basis.

Finally, the *trace information* records the message's route through the MTS. This is used for detecting loops in routing. The first trace component also gives the submission time of the message.

In the 1984 specifications, the P1 protocol was restricted to carrying three types of message. These were the P2 message, the delivery report, and the probe. The P2 message we've met above. The other two are described below.

Delivery report

The *delivery report* is a pseudo-message generated by an MTA which specifies either that the delivery of the message was successful (if the user requests reports of successful deliveries) or that the message failed to be delivered. The first case allows the user to determine if the message was really delivered to the recipient.

continued on next page

The X.400 Message Handling System *(continued)*

The second case allows for delivery failure to be reported. As X.400 labels the type of message, error messages can be handled in a different way to normal messages (e.g. delivery reports are prohibited from creating new delivery reports).

Probe

The third type of message, the *probe*, is a stop gap. It allows users to test out a route to a destination. It then reports back whether or not they can reach the destination with the required attributes. It does a little more than just test the route, allowing users to see if the recipient can receive the format of messages they intend sending. It is therefore just a stand in until the Directory system is running, when this sort of function will be much easier.

P21

The 1988 version of P1 (P21) has few basic functional changes but is now far more complex. The major changes in the protocol are mechanisms for strong security and authentication; a method for extensions; the addition of distribution lists; the provision for message stores and the interaction with other media.

Security

The security mechanisms in X.400 are complex, but briefly the following cases are considered:

Invalid users
Masquerading as other users
Replaying of previously sent messages
Message modification
Traffic analysis
Denial of sending a message
Security level violations
Modification of routing information
Preplay — sending a copy of a waiting message ahead of time

Security is also discussed in the X.500 Directory standards, particularly in the Authentication Framework [4], and depends greatly on concepts used in the RSA algorithms and use of public keys.

Extensions

The 1988 X.400 extension mechanism is designed to solve the lack of extensibility of 1984 X.400. This in itself is a problem, in that if 1984 X.400 is not extensible, how can a superset protocol interwork with it? Put simply, it can't. X.400 1984 and X.400 1988 do *not* interwork. The 1988 version MTAs must attempt to downgrade to the 1984 protocol if possible when gatewaying to 1984 systems. However, certain attributes can not be downgraded or thrown away so some messages will be rejected at such gateways.

The extensions provide ways to add extra information to the envelope. This allows new services to be added and private information to be conveyed between MTAs. Each extension comes with a flag which specifies if this particular extension must be understood for the message to be handled. If so, and the MTA does not understand the extension, the message must be rejected.

Distribution lists

The lack of distribution list support in the 1984 specification has been amended in the 1988 specification. There are now provisions in the protocol for checking that a message expanded by a distribution list is not looping. It is also defined how an MTA should expand a list.

A distribution list has the following properties:

A list of members
A list of users and other distribution lists that can submit to it
An expansion point, which is an MTA that should expand the list
An owner who is responsible for the list management

Delivery reports occurring after the expansion of a list are always sent back to the list, where they are either forwarded to the originator, to the owner of the list, or both. It is intended that the data for the distribution lists will reside in the Directory system.

Message Stores

Message Stores are another addition within the 1988 version of X.400. They allow an MTA to not only deliver to a user agent, but to deliver to a message store. A user agent can then connect to this store at a later time using a specified access protocol (P7) and perform operations on the messages within the store. This function provides a remote mailbox which can be accessed from different places, and might typically be used by a user on a PC with limited storage. The message store also allows many functions and the capability to extend these functions to new formats of messages. The message store operations come in several categories.

Retrieval

Message *retrieval* is perhaps the most obvious operation that a message store should provide. The message store provides operations to summarise or list messages; to fetch particular messages from the store; to delete messages from the store; and to register actions to be automatically undertaken with the store.

The message store allows indirect submission of messages to the MTA. In this case the message is given to the message store which will then undertake to submit the message to the MTA. The message store allows a number of administrative operations to be performed, such as the changing of user credentials.

The selection of messages can be done in a fairly complex fashion. Selections allow message to be selected using ranges such as time or numbers; filters which specifies constraints on attributes and limits or the resulting matches.

Other interfaces

In the 1988 protocol, provision is made to connect the message system to other media, in particular to the postal system and telex systems. This brings in a whole host of new error codes such as:

undelivered-mail-postman-bitten-by-recipients-dog.

It also brings a whole new set of problems, such as how delivery reports make their way back from the mailman to the X.400 system.

One other extension is worthy of note. In the 1984 specification, the only content type that was defined to be carried by P1 was P2. In the 1988 specification, P1 can carry either P2, P22 or an externally defined type. In this way, P1 can carry other protocols and be used for other store and forward applications if required.

Comparison with RFC 822

Presented here is a brief comparison of RFC 822 and X.400. In general, the two systems have the same goals, but X.400 is designed on a wider scale encompassing existing services such as Fax and Telex.

continued on next page

The X.400 Message Handling System *(continued)*

Some of the more significant advantages of X.400 over RFC 822 are:

- Handling of multi-media documents in message.
- Typed messages, allowing error messages to be distinguished from normal messages.
- A structured message format, allowing elements to be defined in a language independent format.
- Conversion of body components where sensible to allow delivery to devices not understanding a given format.
- Interaction with the X.500 Directory services for name lookup.
- Delivery reports are part of the protocol allowing positive and negative confirmations.
- X.400 is an *international standard*. This is by far its biggest benefit. Having a common standard adopted by all countries for Message Handling is a necessity.

Possible defects of the X.400 recommendations are:

- Addressing is complex. This will be solved by future Directory systems, but at present addresses are not very user friendly.
- The protocol is large. X.400 is complex to implement, but the functionality is large—you get what you pay for!

X.400 and RFC 822 interworking

RFC 987 specifies a mapping between the X.400 protocol elements and the RFC 822 Internet standard. This allows for interworking between X.400 and RFC 822 although much of the X.400 functionality is lost in the process.

Conclusions

In conclusion, X.400 mail seems to have great promise and some real advantages over the currently popular RFC 822. I look forward to sending pictures in my messages in a secure manner with confirmation of delivery!

References

[1] ISO Standard "10021 Information Processing systems—Text Communications—MOTIS."

[2] CCITT Recommendation X.400 "Message Handling Systems: System Model—Service Elements."

[3] CCITT Recommendation X.208/ISO Standard 8824 "Specification of Abstract Syntax Notation One (ASN.1)."

[4] CCITT Recommendation X.509/ISO Standard ISO-9594-8 "The Directory—Authentication Framework."

JULIAN ONIONS received his M.Phil in Computer Science from Nottingham University in 1988. For the past 5 years he has worked mainly in the area of message handling as a Research Assistant at Nottingham University. He has been involved in various projects involving distributed applications, standardisation issues and networking. He was involved in the development of the MMDFII RFC 822 based mail system. He is currently part of a group writing a message system for X.400 and RFC 822 mail, PP, and working on a group communication project, COSMOS. He has also contributed to the development of ISODE.

Connectathon '89

A week of testing

More than 65 computer hardware and software manufacturers gathered in Santa Clara, California in mid-February for a week long *Connectathon*—an around-the-clock connectivity marathon that tested interoperability amongst various implementations of the Open Network Computing/Network File System (ONC/NFS™), NeWS™, and X11™. NeWS and X11 are windowing technologies, from Sun Microsystems and the Massachusetts Institute of Technology, respectively. NFS is the de facto industry standard for sharing files across a broad range of computer systems. (NFS has more than 260 licensees worldwide).

The object of Connectathon '89 was to achieve complete interoperability between all the various implementations of NFS and the associated protocols. More than 95 implementations of NFS alone were being tested, resulting in a "test matrix" with over 3000 "check marks."

Many vendors

According to Sun's Vice President of Marketing Ed Zander, the event has grown at a rapid pace since Sun launched it in 1986. He said that the number of participants at this year's Connectathon has grown by one-third since the last event. "The extremely broad support demonstrated here for NFS and network computing underscores the growing commitment to open systems. Open networking computing has become an essential part of any overall computing solution," said Zander.

Window systems

There is a growing interest in windowing systems, and this was reflected in the Connectathon by over 30 implementations of X11 and NeWS. Several new network terminals based on this windowing technology were being shown for the first time during the Connectathon.

Network License Servers: The Next Step Toward Interoperability?

by James H. Geismann
and Thomas P. Wood, Marketshare, Inc.

Introduction

Despite advances in interoperability, end-users still cannot freely access software resources. Often there is no assurance applications will use the most cost-effective hardware resources to process, store or display results. "True" interoperability is still a long way off, if you think of it as transparent access to software and hardware regardless of machine type and operating system.

However, new technology—in the form of *network license servers*—is bridging the gap. In many ways, as described in this article, this technology provides "virtual" interoperability from the end-user's perspective. The technology also has other side benefits in the areas of systems management, license distribution, and pricing flexibility.

User needs driving interoperability

Interoperability is a powerful concept shaped by the needs of end-users. Advances in TCP/IP protocols make it easier to store and coordinate data on and across networks. Users are more productive using TCP/IP assisted data transfer and remote machine access. To achieve full interoperability, the application user must be able to:

- Transfer data between machines, regardless of operating system
- Remotely access any machine from anywhere in a network.
- Conveniently access applications on the network without loss of management control.
- Efficiently use network-wide computing power.
- Experience access and control procedures transparently across different operating environments.

Although the last three user needs have been obvious for a long time, few vendors have addressed interoperability of application software. The primary reason is the technical difficulties of getting software to operate transparently across networks containing hardware platforms from different vendors.

CPU-locked software

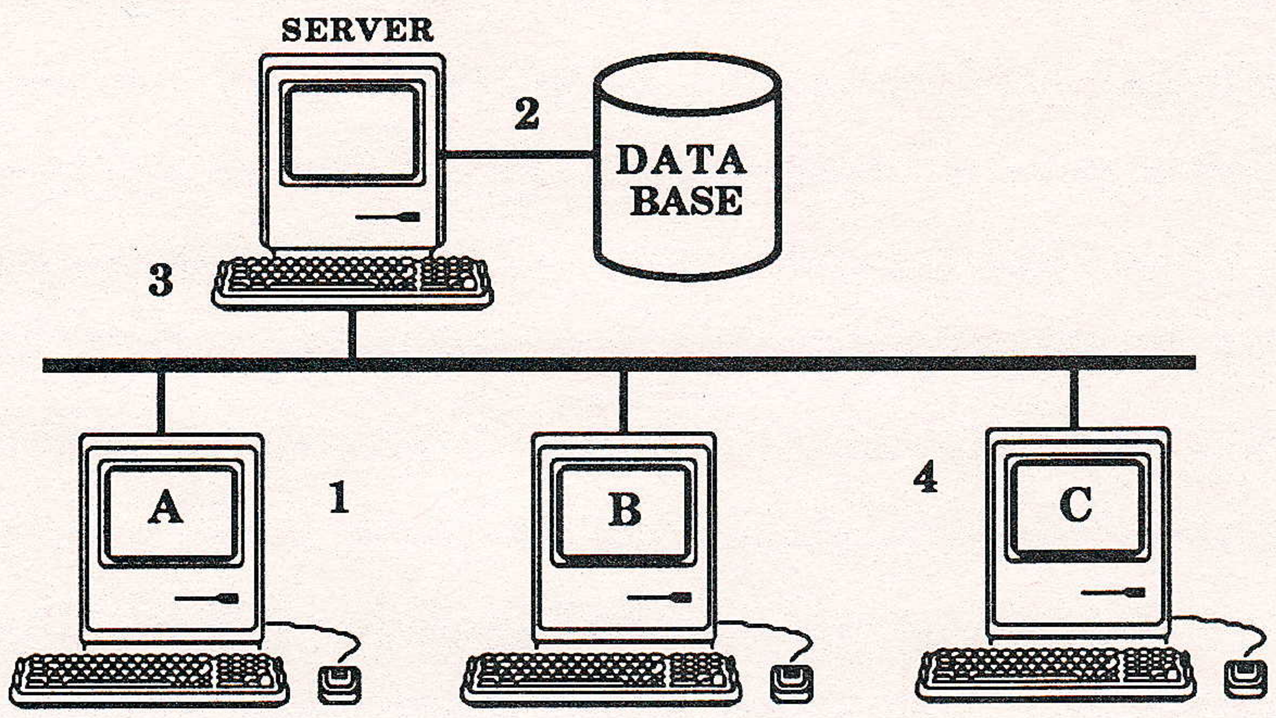
But much of the difficulty comes from the widespread use of CPU locked software as a means of copy protection and user control. (Such methods reflect a throwback to mainframe days when computing was centralized—and life was much simpler). Remote access to CPU-locked software often means a maze of different logons, prompts, and menus. Often, a user winds up using an inappropriate or undesirable CPU.

Sometimes network response times and overburdened remote CPUs cause user dissatisfaction. This often leads to illegal copying to get programs to run locally.

Recent developments like NCS, NFS, X-Windows, et al. are making applications more accessible and compatible. These technologies are a major step toward applications interoperability. But they fail to satisfy the needs of the end-users while protecting the interests of applications vendors.

Network license servers

Network license servers consist of two elements: an interface which is added to new or existing software and an application (often a server) that keeps track of the number of current users and the number of authorized copies in use at one time.



1. Application on A checks server before running
2. Database is checked for no. of simultaneous users
3. "OK" response lets user A run application
4. All licenses active: program will not run

Figure 1: Network License Server Scenario

Before letting a user run an application, the server checks to see whether an additional user will exceed the number of authorized copies available. If this new user remains within the authorized limits to the number of simultaneous users, the application is permitted to run. If the number of simultaneous users is exceeded, the new user is so informed and may be placed in a waiting queue.

Notice the applications with the server interface are no longer copy protected and can be copied throughout the network. The license server controls the number of applications that are running, not the number that exist. Consequently, applications cannot run without the server, ensuring compliance with license agreements.

Architectures

In a recent multi-client project we found at least 18 different server implementations. However, all address the same basic issue of making it easier to access applications on a network. A partial list of companies and servers is given below, E=embedded product, S=standalone product.

Company	Server name	Type
Apollo	Network License Server	S
Computervision	License Manager	S
Connect Computer	Turnstyle	S
Hewlett-Packard	ID Module	S
Integrity Software	Site Lock	S
Auto-trol	License Management	E
Cadre	Teamwork Project Environment	E
DEC	License Management Facility	E
Frame Technology	Floating License Server	E
Intergraph	Intergraph's flexible licensing	E
Valid Logic	ACCESS	E

continued on next page

Network License Servers *(continued)*

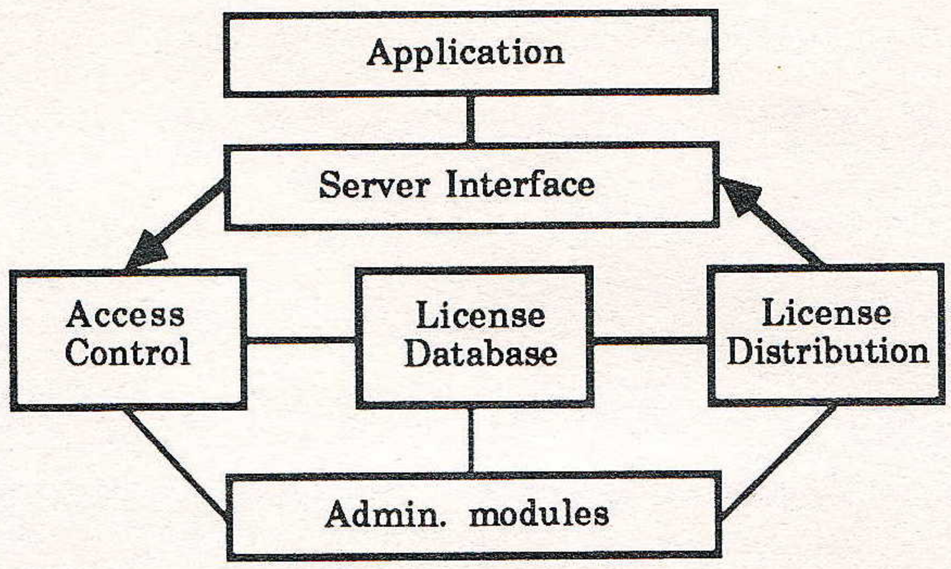


Figure 2: Network License Server— Basic building blocks

We also found the different implementations seem to use a common set of building blocks (see Figure 2):

- *Server Interface:* This module requests licenses or applications from the server.
- *Access Control:* This module contains information on who can access the licenses. Node-ID, user-ID, and time are just some ways of controlling access.
- *License Database:* This module contains information about applications that are controlled by the server. The number of purchased licenses resides here.
- *License Distribution:* This module is responsible for sending the license to the correct application on the correct CPU.
- *Administration:* This module monitors license usage and related transactions as applications are accessed and licenses are checked in and out. This module may contain utilities for adding licenses or changing access information.

Despite the commonality, there are differences. Some implementations can be used across multiple platforms and operating systems. Others offer elaborate license security and features that make control and distribution of software easier for the end-user.

Virtual interoperability

One might almost say network license servers can make applications become "virtually" interoperable. Indeed applications are not really interoperable because they cannot magically run on any machine. But from a user's perspective the applications can look that way providing:

1) the vendor has ported the application to different hardware platforms, and

2) the server keeps track of total number of simultaneous users of an applications regardless of machine or operating system.

Useful technology, but not a panacea

Network license servers can be used with either commercially available or proprietary, in-house applications. This technology lets applications vendors offer more flexible licensing arrangements, access, and pricing. End users can use license servers to access and control software according to their in-house usage patterns.

Network license servers offer benefits to both end-user and vendor. But as the song goes "freedom has its price."

End-users may face these changes from applications using network license servers:

- More convenient access to applications on the network may be offset by increased challenges to systems management.
- Software that uses specialized hardware may not be helped by this technology.
- Software management is easier, more centralized and may result in new administrative procedures or controls.
- In-house proprietary software may be more closely controlled (and therefore monitored).
- Software utilization rates will rise so some companies may acquire fewer software licenses. The waiting times for hardware access may now be replaced by queuing up for software.
- Unit prices for software may go down, but total software costs will rise as more people seek access.

Vendors may face some interesting challenges as they sell software using network license server schemes:

- Software demand may increase at some sites while drying up at others. Sales and support requirements will be hard to forecast.
- Some customers will recognize the value of improved access and will pay for it. Others will see this as a free ride and will be dissatisfied because they mis-estimated software demand.
- Competition may be based in innovative access, rental or pricing schemes. Vendors will need to rethink the value received by their customers.
- Risk of less sales, coupled with new pricing opportunities will make it harder to cover R&D expenses.

By and large, end users will face management issues with license servers. Application vendors will face severe marketing challenges.

Conclusions

Users need interoperability to make better use of current hardware and software. Before network license servers, meeting the needs of end-users for software interoperability was viewed as a threat by software vendors. Often user demands were not met because of technical difficulty. Now the only excuse for not meeting these needs is lack of business acumen or foresight.

One senior executive called Network License Servers "a bad idea whose time has come." Well, good or bad, now is the time to use license servers to enhance interoperability.

JAMES H. GEISMANN, president, and **THOMAS P. WOOD**, consultant, are with Marketshare, Inc. (Wayland, MA,) a business development and marketing consulting firm that assists clients offering high-technology, computer-related products. The information in this article draws on the results of a recently completed project report on the impact of network license servers on vendors and end-users of technical software.

Upcoming Events

INARC Workshop

A *Workshop on The Future of the Internet Architecture and Protocols* will be held June 1-2 1989 at the University of Delaware. The workshop is sponsored by the Internet Architecture Task Force (INARC) and the Internet Activities Board (IAB).

The IAB has been guiding and coordinating the research and development activities of the DARPA/NSF Internet System for several years. The Internet Architecture Task Force of the IAB has been asked to explore the inherent limitations in the existing Internet architecture and supporting TCP/IP protocol suite and how the lessons learned can be applied to future systems. The two-day workshop will explore these and related issues. While the emphasis of the workshop will be on the past and future evolution of the Internet system, specific issues relevant to other architectures, protocol suites and migration strategies may be discussed as well.

Who should attend?

Interested persons from all walks of network life are invited to attend. Participants will be encouraged to present short briefings on specific technical issues, including those suggested below, but this is not a requirement for admission. While some participants may be invited on the basis of their known expertise, biases and past vocalizations on these issues, participants outside the IAB, INARC and their dependencies are actively encouraged. In order to manage the local arrangements it is necessary that participants register their intent to attend by contacting the INARC chair:

David L. Mills
Electrical Engineering Department
University of Delaware
Newark, DE 19716
302-451-8247 or mills@udel.edu

Topics

Areas of interest include, but are not limited to, the following:

- Are the Internet architecture and protocols suitable for use on very high-speed networks operating in the 1000 Mbps range and up? If the network-level or transport-level protocols are not usable directly, can they be modified or new ones developed to operate effectively at these speeds?

- Are the Internet addressing and gateway-routing algorithms adequate for very large networks with millions of subscribers? If not, is it possible to extend the addressing scope and/or develop new routing paradigms without starting over from scratch?

- Can the Internet model of stateless networks and stateful hosts be evolved to include sophisticated algorithms for flow management, congestion control and effective use of multiple, prioritized paths? Can this be done without abandoning the estimated 60,000 hosts and 700 networks now gatewayed in the system?

- Can the existing Internet of about 300 routing domains be evolved to support the policy and engineering mechanisms for many thousands of domains including education, research, commercial and government interests? Can this be done with existing decentralized management styles and funding sources? If not, what changes are needed and how can they be supported, given practical limits on infrastructure funding?

ACM SIGCOMM '89

The *ACM SIGCOMM '89 Symposium on Communications Architectures and Protocols* will be held September 20-22, 1989 at the University of Texas at Austin. Tutorials will be held on September 19th.

The symposium provides an international forum for the presentation and discussion of communication network applications and technologies, as well as recent advances and proposals on communication architectures, protocols, algorithms and performance models.

Topics

The areas of interest for the symposium include, but are not limited to the following:

- Analysis and design of computer network architectures and algorithms,
- Innovative results in local area networks,
- Computer-supported collaborative work,
- Network interconnection and mixed-media networks,
- High-speed networks,
- Resource sharing in distributed systems,
- Distributed operating systems and databases,
- Protocol specification, verification, and analysis.

Proceedings

The Proceedings will be distributed at the symposium and published as a special issue of ACM SIGCOMM *Computer Communication Review*. A few of the submitted papers will be selected for publication in the ACM *Transactions on Computer Systems*. For more information about the symposium, contact:

Dr. L. H. Landweber, General Chair
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
Telephone: 608-262-1204
EMAIL: landweber@cs.wisc.edu.

Internetworking tutorials

Advanced Computing Environments will offer the *TCP/IP OSI Internetworking Tutorials*, June 19–22 in Dallas. Topics include an In-Depth Introduction to TCP/IP, Berkeley UNIX Networking, TCP/IP for the VM Systems Programmer, Local Area Networks and TCP/IP Alternatives, Message Handling and Directory Systems—X.400/X.500, The Domain Name System, Bridges and Routers, Network Management, Network Operations and Security, and Practical Perspectives on OSI Networking.

For a complete tutorial program, call Advanced Computing Environments at 415-941-3399.

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040
415-941-3399
FAX: 415-949-1779

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1